Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Symbolic Computation 37 (2004) 641–668
Journal of Symbolic Computation
www.elsevier.com/locate/jsc

A relative van Hoeij algorithm over number fields

Karim Belabas*

Département de mathématiques, Université Paris XI, F-91405 Orsay, France

Received 29 April 2003; accepted 16 September 2003

Abstract

van Hoeij's algorithm for factoring univariate polynomials over the rational integers rests on the same principle as the Berlekamp–Zassenhaus algorithm, but uses lattice basis reduction to improve drastically on the recombination phase. His ideas give rise to a collection of algorithms, differing greatly in their efficiency. We present two deterministic variants, one of which achieves excellent overall performance. We then generalize these ideas to factor polynomials over number fields.

MSC (1991): 11Y40; 11Y05

Keywords: Polynomial factorization; Number fields

0. Introduction

Until 2000, the two main algorithms able to factor a polynomial P over $\mathbb{Q}[X]$ were the Berlekamp–Zassenhaus algorithm (Berlekamp, 1970; Zassenhaus, 1969), which starts by factoring P over $\mathbb{Q}_p[X]$ for a suitable prime number p and tries to recombine the p-adic factors, and the Lenstra–Lenstra–Lovàsz algorithm, based on their celebrated LLL lattice reduction algorithm (Lenstra et al., 1982). While the latter is polynomial-time and the former exponential in the worst case[1], the former performs far better on average, in practice.

Lately, van Hoeij (2002) published an algorithm which, while following the Berlekamp–Zassenhaus argument, uses lattice basis reduction to guess the correct

* Tel.: +33-1-69-15-57-48; fax: +33-1-69-15-60-19.
E-mail address: Karim.Belabas@math.u-psud.fr (K. Belabas).

[1] An irreducible polynomial P can have as much as $\Omega(\deg(P))$ factors over $\mathbb{Q}_p$, so recombination takes time $2^{\Omega(\deg(P))}$.

0747-7171/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsc.2003.09.003

recombination. The main idea is as follows: assume that P is integral and monic; then Newton sums of an integral factor of P are easily shown to be bounded integers. Provided the p-adic accuracy p^a is large enough, they are small compared to the Newton sums of non-integral modular factors lifted from $\mathbb{Z}/p^a$ to $\mathbb{Z}$. Finding a valid recombination is thus reduced to finding simultaneous small values of linear forms with integer coefficients, or alternatively solving a knapsack problem, one of the very situations where the LLL algorithm is known to be of interest. At the time of this writing, none of the algorithms derived from van Hoeij's ideas has been proven to run in polynomial time. On the other hand, after three years of experimentation, no practical bad case is known either.

In the first section, we recall the details of van Hoeij's algorithm over $\mathbb{Q}$. The second section describes tunings and our experiments with this algorithm. The third section presents a generalization to number fields.

Our implementations are part of the PARI library (The PARI group, 2003). All timings were obtained with PARI-2.2.6 configured to use GMP-4.1 as its multiprecision kernel, on a 1 GHz Athlon under Linux (lucrezia.medicis.polytechnique.fr), and are given in seconds.

1. Details of van Hoeij's method

For the basic terminology for the LLL algorithm we refer the reader to Cohen (1993) and Pohst and Zassenhaus (1989). The term *quality ratio* denotes the real number $1/4 < \alpha \leq 1$ used to check the Lovàsz condition $B_i \geq (\alpha - \mu_{i,i-1}^2)B_{i-1}$, which determines the frequency of swaps in LLL; higher quality ratios mean smaller basis vectors, at the expense of higher running times. We let $\gamma(\alpha) := (\alpha - 1/4)^{-1} \geq 4/3$. We refer the reader to van Hoeij's original paper (van Hoeij, 2002) for further details and examples.

1.1. The knapsack

Let $\mathbb{K}$ be a field and $G \in \mathbb{K}[X]$ a polynomial. Its kth Newton sum is $S_k(G) := \sum_l \alpha_l^k$, where the α_l run through the roots of G in an algebraic closure of $\mathbb{K}$, repeated according to their multiplicity. It follows that $S_k(GH) = S_k(G) + S_k(H)$, for $G, H \in \mathbb{K}[X]$. Furthermore, as a symmetric function of the roots, $S_k(G)$ belongs to $\mathbb{K}$. If G is monic with algebraic integer coefficients, then $S_k(G)$ is an algebraic integer. If G has complex coefficients, let $B_{\text{root}}(G)$ be an upper bound for the modulus of the roots of G; then

$$|S_k(G)| \leq \deg G \cdot B_{\text{root}}(G)^k. \tag{1}$$

Let $P \in \mathbb{Z}[X]$, which we assume monic for simplicity. Let p be a prime not dividing the discriminant $\text{disc}(P)$ of P, and let (G_i), $1 \leq i \leq n$, be the monic irreducible p-adic factors of P in $\mathbb{Z}_p[X]$. We are looking for the vectors (ε_i) such that $P_\varepsilon := \prod_i G_i^{\varepsilon_i}$ has integer coefficients. If the coefficients of the G_i are known to the precision p^a, i.e. modulo $p^a\mathbb{Z}_p$, this means that the coefficients of $\prod_i G_i^{\varepsilon_i}$ do not vary with a when a is large enough. As a consequence, the Newton sums $S_k(G_\varepsilon) = \sum_i \varepsilon_i S_k(G_i)$ are close to a multiple of p^a. More precisely, we are looking for $\{0, 1\}$ vectors (ε_i) such that

$$\sum_{i=1}^{n} \varepsilon_i S_k(G_i) + \lambda p^a + \mu = 0, \tag{2}$$

for a small μ which is estimated from Eq. (1). (λ is likewise estimated; this estimation is not needed in the absolute set-up, but is required over number fields.) Identity (2) is a knapsack problem and the set of rational integer vectors (ε_i) providing a solution of Eq. (2) for all k is a sublattice $L_{\mathbb{Q}}$ of $\mathbb{Z}^n$. Solving our initial problem amounts to finding a basis of this sublattice.

1.2. Solving the knapsack problem

To achieve this, van Hoeij applies an iterative process. Let

$$M = \begin{pmatrix} C\mathrm{Id}_n & 0 \\ S & p^a \mathrm{Id}_N \end{pmatrix},$$

$$\text{where } S \text{ is a lift of } \begin{pmatrix} S_{i_1}G_1 & \cdots & S_{i_1}G_n \\ \vdots & \ddots & \vdots \\ S_{i_N}G_1 & \cdots & S_{i_N}G_n \end{pmatrix} \bmod p^a, \tag{3}$$

$\{i_1, \ldots, i_N\} \subset \{1, \ldots, n\}$ and $C \geq 1$ is a suitable integer constant. Let q be the standard Euclidean form over $\mathbb{R}^{n+N}$. Then we are looking for vectors of the lattice $(\mathrm{Im}\ M, q)$ of norm smaller than some bound B. We now compute an LLL-reduced basis of this lattice, and discard part of this basis by using the following classical lemma:

Lemma 1.1. *Let (b_i) be a basis of the lattice (Λ, q) and let (b_i^*) be its Gram–Schmidt orthogonalization. Let $B \geq 0$ be such that $q(b_j^*) > B$ for $j > j_0$. If $v \in \Lambda$ satisfies $q(v) \leq B$, then v belongs to the subspace generated by $(b_1, \ldots, b_{j_0})$.*

Using an LLL-reduced basis guarantees that b_i and b_i^* have roughly the same size and that the size of the b_i^* does not decrease too fast, so we may in fact expect that the last ones have large norm. More precisely, let α be the chosen quality ratio and $\gamma := (\alpha - 1/4)^{-1} \geq 4/3$; then we have

$$q(b_j^*) \leq q(b_j) \leq \gamma^{i-1} q(b_i^*), \qquad \text{for all } 1 \leq j \leq i.$$

After a number of such reduction/elimination steps, let $L \supset L_{\mathbb{Q}}$ be our lattice (initially, $L = \mathbb{Z}^n$). If $L = L_{\mathbb{Q}}$, then its Hermite normal form (HNF) basis satisfies the condition:

$$\text{it has entries in } \{0, 1\}, \text{ with a single 1 in each row.} \tag{4}$$

To check whether we are done, we compute the HNF basis of L. If condition (4) is satisfied, we presumably have found a factorization of P, which is checked as in the usual Berlekamp–Zassenhaus process: multiply the factors, lift the result, and finally try to divide P by the putative factors after various preliminary, less expensive, checks, such as the "$d - k$ tests" of Section 2. If all divisions are exact, then $L = L_{\mathbb{Q}}$; the factors are irreducible since they correspond to a basis of $L_{\mathbb{Q}}$.

1.3. Truncation

A major practical improvement, also proposed by van Hoeij, is to replace p-adic values $S_k(G_i)$ truncated to precision p^a by twice-truncated integer approximations $T^{a,b}(S_k(G_i))$,

defined by

$$T^{a,b}(x) := \frac{x - (x \bmod p^b)}{p^b} \bmod p^{a-b},$$

where x is known to precision p^a and $b < a$. Note that $x \bmod y$ is to be understood as: the integer congruent to x modulo y which belongs to the interval $]-y/2, y/2]$. In effect, instead of looking for linear combinations smaller than some bound, we find combinations of leading p-adic digits which are close to 0:

Lemma 1.2 (van Hoeij, 2002, Lemma 2.6). *Assume $|S_k(G)| < p^b/2$ for any divisor G of P in $\mathbb{Z}[X]$, and let $a \geq b$. If $G := \prod_{i\in I} G_i \in \mathbb{Z}[X]$ for some subset I of $\{1, \ldots, n\}$, then $T^{a,b}(S_k(G)) = 0$ and*

$$\left| \sum_{i\in I} T^{a,b} S_k(G_i) \right| \leq |I|/2 \leq n/2.$$

The bound is independent of p, a and b; it only depends on the number of modular factors. The lattice to be LLL-reduced has entries of the order of p^{a-b} where $b < a$ is essentially a *free* parameter, speeding up reduction as it increases. The downside is that we do not exploit fully the system rigidity since we use a subset of the available p-adic digits, possibly requiring more LLL reductions.

1.4. Comments

The precise choice of the matrix M depends on the choice of the subset of Newton sums (S_{i_k}). Repeating this process while varying the Newton sums and/or the precision b, which can be set independently for different sums, is expected to decrease quickly the size of the lattice under consideration. Unfortunately, we cannot prove this, nor that the algorithm terminates when specified in this form. (van Hoeij proves that his algorithm must terminate as the p-adic precision a goes to infinity, but does not give an explicit bound.) Aborting the sequence of LLL reductions and finishing by an exhaustive enumeration of the lattice small vectors is a theoretical solution, but amounts to (part of) a naïve recombination, and is exponential in the worst case.

2. Tuning van Hoeij's algorithm

In this section, we describe two variants of van Hoeij's algorithm which we have experimented with. The first one, originating in Belabas et al. (2001), turns out to be unsatisfactory in large dimensions, while we found the second to be very efficient. As the original algorithm, both variants are heuristic. Although we cannot even prove it terminates in the form specified below, we conjecture the second one runs in polynomial time (see Remark 2.5).

2.1. The "$d-k$" tests

van Hoeij's idea can be used to speed up naïve recombination, and is related to the $d-1$ and $d-2$ tests of Abbott et al. (2000), though easier to implement and less memory-hungry.

In fact, it provides an efficient "$d-k$ test"[2] for any $k \geq 1$: assume the modular factors are lifted to precision p^a, and let $b < a$ be minimal such that

- $|S_k(G)| < p^b/2$ for any divisor G of P.
- (for efficiency) p^{a-b} can be represented as a machine integer.

Precompute $s_{i,k} := T^{a,b}(S_k(G_i))$ for all modular factors G_i. Using Lemma 1.2, check that

$$\left| \sum_{i \in I} s_{i,k} \bmod p^{a-b} \right| \leq |I|/2$$

before trying the putative factor $\prod_{i \in I} G_i$.

2.2. Root bounds

Let ρ be the largest modulus of the roots of P, and B_{root} be an upper bound for ρ. In van Hoeij's method, the bound Eq. (1) increases as B_{root}^k with the trace index k, in contrast to the fixed factor bound in Zassenhaus's algorithm. So it is important to make the most of lower degree traces, as we shall in Section 2.4, and to derive sharp bounds for ρ.

Let $P = a_0 X^h + \cdots + a_h$, where $a_0 a_h \neq 0$; Cauchy's method bounds ρ by the unique positive root of

$$|a_0| X^h - \sum_{i<h} |a_{h-i}| X^i$$

which is easy to approximate by dichotomy, and is sharp up to a factor $2^{1/h} - 1$, where $P = (X+\rho)^h$ realizes the worst case (see Birkhoff, 1914). All a priori bounds we are aware of (see e.g. Stoer and Bulirsch, 2002; Householder, 1970; Knuth, 1969, 4.6.2—Exercises 19, 20) use the absolute values of the coefficients a_i, and hence are at least as large as Cauchy's bound.

Improved bounds are obtained by using a few Graeffe iterations before using Cauchy's method, as suggested in Cerlienco et al. (1987) and Davenport and Mignotte (1990). In fact, arbitrarily tight bounds may be obtained in this way (see Gourdon, 1993), but these are expensive.

2.3. Strategy A: decreasing dimension

In this section, we build a decreasing sequence of lattices $L_k \supset L_{\mathbb{Q}}$, embedded in $\mathbb{Z}^{r_k}$ where $r_k := \dim L_k$, instead of remaining embedded into $\mathbb{Z}^n$. The dimension of the matrices fed to LLL decreases rapidly (it is dimension $r_k + N$ instead of remaining at dimension $n + N$), which induces faster LLL reductions. The lattice L_k is represented by a basis, given by the columns of a matrix M_{L_k}.

[2] The $d-k$ tests defined in Abbott et al. (2000) for $k = 1, 2$ check the *coefficient* of degree $d-k$ of a potential factor of degree d. van Hoeij's test checks the Newton sums instead of the symmetric functions of the roots. It is linear in the modular factors.

For a $m \times n$ matrix $M = (m_{i,j})$ with real coefficients, we write

$$\|M\|_2 := \left(\sum_{i,j} |m_{i,j}|^2 \right)^{1/2}$$

and define an ad hoc norm by $\sup_{x \in \{0,1\}^n} \|Mx\|_2$. The square of this norm is bounded by $n\|M\|_2^2$, but more sharply by

$$\|M\|^2 := \sum_i \max\left(\left| \sum_{j, m_{i,j} > 0} m_{i,j} \right|^2, \left| \sum_{j, m_{i,j} < 0} m_{i,j} \right|^2 \right).$$

In the description below, the subscript k has been dropped, and we assume the p-adic truncation $T^{a,b}$ operates coordinatewise.

Algorithm 2.1 (SearchTrueFactors: Project Lattice). Input: a list of n modular factors known to precision p^a, positive integers *tried* and N. Assume all combinations with (strictly) less than *tried* factors have already been tried. N is the number of new traces to include in each run.

Output: the factors over the rationals.

Initialization: Let $s = N$, $r = n$, M_L the identity $n \times n$ matrix, B_{root} a bound for the modulus of the largest complex root of P.

For $k = 1, \ldots$ repeat the following steps:

(1) Compute $B_{\text{trace}} := \deg(P) B_{\text{root}}^s$ an upper bound for $S_k(P_\varepsilon)$ if $P_\varepsilon \mid P$ in $\mathbb{Q}[X]$ and $k \le s$.

(2) [Invert M_L (n by r matrix)]: Using a modular algorithm, compute a left inverse M_L^{-1} of M_L.

(3) [Compute bound for $B_i = |b_i^*|^2$]: Let

$$C := \left\lceil \sqrt{Nn^2/(4\|M_L^{-1}\|^2)} \right\rceil, \qquad \text{and} \qquad B_{\text{lat}} := C^2 \|M_L^{-1}\|^2 + Nn^2/4.$$

(4) [Build the LLL input matrix]: Let S_0 be the $N \times n$ matrix of traces of degree $s - N + 1, \ldots, s$. Then let M be an $(r + N)$ square matrix, as in (3), where C is as above and $S := T^{a,b}(S_0 \times M_L)$ (of dimension $N \times r$). In other words, S gives the truncated traces of the p-adic polynomials which form our basis for L. See below for how to choose b.

(5) [Lattice reduction]: LLL-reduce in place the $(r + N)$ by $(r + N)$ matrix M, of rank $r + N$. As a by-product, we obtain the norms of the Gram–Schmidt vectors associated with the LLL-reduced basis: let B_i, $1 \le i \le r + N$, be the squared norm of the ith orthogonal vector.

(6) Let r' be the smallest index such that $B_i > B_{\text{lat}}$ for all $i > r'$. If $r' = 1$, return "irreducible".

(7) [Update M_L]: Let U be the $r \times r'$ matrix whose entries are those of the upper left part of the new M, divided by C (exact division). If U does not have maximal rank, replace it by its integral image (the non-zero columns of its HNF).

(8) Replace M_L by $M_L \times U$; M_L is now of dimension $n \times r'$, of rank r'.

(9) If $r' > n/tried$, go to step (11).

(10) [Check]: Put M_L in HNF. If condition (4) is satisfied, check whether M_L yields a valid factorization as in Berlekamp–Zassenhaus. If so, abort the algorithm and return the factors.

(11) Increase s by N, replace r by r'.

(12) If k gets too large, abort the algorithm and switch to Berlekamp–Zassenhaus to finish the recombination.

Proof. First M_L admits a left inverse in Step (2), since it has maximal rank by induction. It remains to prove that, in Step (5), to each true rational factor there corresponds a vector x such that $|Mx|^2 \leq B_{\text{lat}}$. As in van Hoeij's paper, we know that to each rational factor there correspond $v \in \{0,1\}^n \cap L_{\mathbb{Q}}$ and $w \in \mathbb{Z}^N$ such that $|S_0 v + p^{b-a} w|^2 < Nn^2/4$. Since $L_{\mathbb{Q}} \subset L$, $v = M_L v'$ for $v' \in \mathbb{Z}^r$ and in fact $v' = M_L^{-1} v$, which does not depend on the chosen left inverse. We choose for x the block vector $\binom{v'}{w}$, and we obtain $|Mx|^2 \leq B_{\text{lat}}$.

The validity of the algorithm now follows as in van Hoeij's argument. Note that we only include the last step to ensure that the algorithm terminates: it is expected that $\|M_L^{-1}\|$ remains small in Step (2), and that we eventually find sufficiently small vectors in Step (5). □

We used the following strategy for the values of (a, b): a is the p-adic precision to which the factors are known, and b is implicitly defined by the parameter $\delta := \log_p \|(S_0 \times M_L) \bmod p^a\|_\infty - b$.

- The initial value p^δ is chosen to have about three times as many bits as the number n of modular factors (at least 32 in any case).
- If $p^b < 2B_{\text{trace}}$, then we increase a and lift all factors and cached Newton sums to a higher accuracy.
- If the new rank r' of the matrix M_L is not smaller than r, increase δ so that p^δ has two more bits per modular factor.

This algorithm has a major drawback: if $\|M_L^{-1}\|$ increases, we are helpless. We can try and find a better inverse, but this involves further LLL reduction and becomes very slow. This unfortunate situation actually occurs when the dimension gets large or M_L diverges from some "optimal" path. Hence the need to stick to relatively high accuracy: at least 3 bits per factor as described above. This value was determined experimentally so as to perform sensibly for $n \leq 128$, say. It can be reduced if n is smaller; and, unfortunately, has to be increased if it gets larger. Even so, for larger n this variant is not practical: our implementation never succeeded in factoring a polynomial with more than 512 modular factors and few true factors.

As proposed by van Hoeij, it is possible to compute different traces to different p-adic precision, or to vary N, the number of traces added in each step, or to use small random linear combinations of traces. But it does not really improve the situation here: the precision has to be kept high to prevent $\|M_L^{-1}\|$ from increasing.

2.4. Strategy B: decreasing b

In this variant, we apply the p-adic truncation operator $T^{a,b}$ to the lattice basis, LLL-reduce the approximate basis and apply the LLL base change matrix to the original basis. We then iterate the process on better and better approximations of that *same* lattice, whose bases are closer and closer to being LLL-reduced. In this process, the parameter b decreases until p^b drops below $2B_{\text{trace}}$. In effect, it is as if we had performed a single LLL reduction on the lattice without applying the truncation operator at all. Since we add p-adic digits incrementally, the bulk of the reduction work operates on small integers.

This is an exact version of Schnorr–Euchner floating point LLL reduction (Schnorr and Euchner, 1994), which should perform particularly well on our class of lattices. That is, short vectors are actually short, with very small entries when expressed in the initial basis, and hence they should be detected in the lattice built from the higher digits of the basis vectors. Assuming the parameter b decreases by a fixed amount δ, we expect fewer LLL runs than the worst case $(a - b_{\min})/\delta$ before finding a truly LLL-reduced basis, and that entries collapse as soon as we find it. In fact, instead of mechanically letting b run through the arithmetic progression $a - k\delta$, we update b in terms of the current input lattice basis, so that δ significant p-adic digits are fed to the algorithm, without "leading zeros".

This is a special case of van Hoeij's method, designed to avoid precision guesses which were the downfall of Strategy A. Low values of δ can be reliably used: if we fail to discard extraneous basis vectors in the initial LLL run because of insufficient reduction, we detect them later, using the *same* traces. Since we restart from an already somewhat reduced basis, we expect smaller entries. Also, for a given group of traces, we can use low quality ratios in all LLL runs but the last, when $b = b_{\min}$, speeding up the intermediary reductions. In short, due to its stability, Strategy B takes full advantage of each computed trace, without having to pay for it.

Algorithm 2.2 (SearchTrueFactors: Decreasing b). Input: a list of n modular factors known to precision p^a, positive integers *tried* and N, a positive real number `BitsPerFactor`. Assume all combinations with (strictly) less than *tried* factors have already been tried. N is the number of new traces to include in each run. `BitsPerFactor` determines the amount of information fed to each LLL run

Initialization: Let $s := N$, $C := \lceil \sqrt{Nn}/4 \rceil$, $CM_L := C$ times the $n \times n$ identity matrix, B_{root} a bound for the modulus of the largest complex root of P, and $B_{\text{lat}} := C^2 n + Nn^2/4$.

Repeat the following steps:

(1) Compute $B_{\text{trace}} := nB_{\text{root}}^s$. Let $b_{\min} := \lceil \log_p(2B_{\text{trace}}) \rceil$.
(2) If $a \leq b_{\min}$, increase a and lift all modular factors and cached Newton traces.
(3) Choose a p-adic precision and build the initial LLL input matrix M, as per Subpart 2.3.
(4) [Lattice reduction]: LLL-reduce in place the matrix M.
(5) Let r be the smallest index such that $B_i > B_{\text{lat}}$ for all $i > r$. If $r = 1$, return "irreducible".
(6) [Update CM_L]: Let CM_L be the $n \times r$ matrix whose entries are those of the upper left part of the new M (after the LLL reduction).

(7) [Iterate]: If $b > b_{\min}$, decrease b and update M as per Subpart 2.4, then go to Step (4). Otherwise, continue to Step (8).
(8) If CM_L does not have maximal rank, replace it by its integral image. Finish with this group of N traces as in Strategy A, and return to Step (1) to feed more traces.

Subpart 2.3 (Initialize M). This sets up the knapsack lattice (caches Newton sums and computes the initial basis) and outputs a suitably truncated basis:

(1) Set $M_L := CM_L/C$ (exact division), $r := \dim M_L$ (upper bound for the number of true factors). Let S_0 be the $N \times n$ matrix of traces of degree $s - N + 1, \ldots, s$. Set $S_1 := (S_0 \times M_L) \bmod p^a$.
(2) [Choose the p-adic precision]: Set δ so that

$$p^\delta \approx \|S_1\|_\infty / 2^{r \cdot \texttt{BitsPerFactor}}.$$

Set $b := a - \delta$. If $b < b_{\min}$, set $b = b_{\min}$ and $\delta := a - b$. The parameter δ is the number of new p-adic digits we input to each LLL run. It is chosen so that we input roughly `BitsPerFactor` bits of information per modular factor.
(3) Let M be the $(n + N)$ square matrix

$$\begin{pmatrix} CM_L & 0 \\ T^{a,b}(S_1) & p^{a-b}\mathrm{Id}_N \end{pmatrix}.$$

Subpart 2.4 (Update M). This computes a suitable truncation of the new knapsack lattice basis:

(1) Set $M_L := CM_L/C$ (exact division), and $S_1 := (S_0 \times M_L) \bmod p^a$.
(2) Set $b := \min(b, \lceil \log_p \|S_1\|_\infty \rceil)$. Decrease b by δ; if $b < b_{\min}$, set $b := b_{\min}$.
(3) Let M be the $(n + N) \times r$ matrix

$$\begin{pmatrix} CM_L \\ T^{a,b}(S_1) \end{pmatrix}.$$

Remark 2.5. As specified above, Algorithm 2.2 may not terminate. To make it rigorous, one may add the following statement to Step (8): if $s > n$, increase the p-adic precision a and restart. Termination now follows from van Hoeij's proof (van Hoeij, 2002, Lemma 2.10).

We conjecture that parameters a and N polynomially bounded in terms of the input size can be chosen so as to guarantee that the algorithm terminates in a single LLL run on the non-truncated lattice (with $b = 0$). It would then be easy to turn Algorithm 2.2 into a rigorous polynomial time algorithm. This variant would remain practical by providing the theoretically required information incrementally and decomposing this huge LLL reduction into many smaller ones: introducing a decreasing truncation parameter b as above, but also increasing successively N then a (initially chosen according to heuristics, not to worst case bounds). We would expect the process to stop long before the theoretical bounds are reached.

Remark 2.6. The original factorization algorithm of Lenstra et al. (1982) admits analogous practical variants. It is obviously possible to decompose its single lattice basis

reduction into many smaller ones, in particular by using lower p-adic accuracies and feeding p-adic digits incrementally as above. This may detect factors at low precision. On the other hand, it seems likely that van Hoeij's algorithm will succeed sooner: it looks for a base change matrix with $\{0, 1\}$ coefficients, whereas Lenstra et al. (1982) look for large coefficients of a rational factor and will need to reach the theoretical bounds before yielding a proven irreducibility result for instance.

2.5. Experiments

2.5.1. Tunings

Strategy A, for those polynomials where it succeeds, was consistently much slower than Strategy B, once good parameters for B had been chosen. Mixing strategies and composing incremental reduction and dimension decrease became pointless since low precision LLL runs become cheap as soon as the lattice rank really decreases (see Section 2.5.3). So we now focus on strategy B.

We use the tuning parameters

$$N = 1 \qquad \text{and} \qquad \texttt{BitsPerFactor} = 0.5$$

which performed best on average. Other values of `BitsPerFactor` between 0.25 and 0.75 give only slightly worse timings (about 20% variation); increasing N, on the other hand, results in much slower LLL runs. In the following experiments, we disabled the "power tests" that speed up the factorization of $P(X^m)$ by recursively factoring P, then $Q(X^k)$ for certain $Q \mid P$ and $k \mid m$. The general strategy is to first try a naïve recombination of all sets of factors with three elements or fewer (our examples were chosen so that it never succeeds), then switch to van Hoeij's method.

2.5.2. Timings

Columns Deg, Dig and n give respectively the degree of the corresponding polynomial, the number of decimal digits of its largest coefficient, and the number of p-adic factors; columns Lift, Knap give respectively the set-up time—small prime factorization and Hensel lifting—and the total recombination time using Strategy B.

Polynomial	Deg	Dig	n	mod p	Lift	Knap	Total
P_4	462	756	42	2.2	13.8	0.4	16.5
P_5	64	40	32	0.0	0.1	0.1	0.2
P_7	384	57	76	1.2	1.6	1.7	4.5
P_8	972	213	54	14.6	15.4	0.6	30.7
$M_{12,5}$	792	2813	72	14.7	65.6	16	96.5
$M_{12,6}$	924	3937	84	26.7	131	76	208
S_7	128	87	64	0.1	0.3	1.2	1.7
S_8	256	188	128	0.6	1.4	26	28.5
S_9	512	402	256	4.6	8.5	718	731
S_{10}	1024	854	512	40.3	78.2	30 065	30 186
$S_6 S_7$	192	127	96	0.3	0.7	7.2	8.2
$S_7 S_9$	640	490	320	8.4	15.2	2 268	2 292
$S_8 S_9$	768	590	384	14.1	27.8	5 867	5 910

The polynomials P_4 to P_8 come from Zimmermann (1996); P_4 has two factors of degree 66 and 396, P_5 to P_8 are irreducible. $M_{12,5}$ and $M_{12,6}$ are the fifth and sixth resolvents of the polynomial f with Galois group M_{12} from van Hoeij's paper (van Hoeij, 2002, Section 3.2); $M_{12,5}$ is irreducible whereas $M_{12,6}$ has two factors of degree 132 and 792; these polynomials are non-monic. S_t denotes the Swinnerton–Dyer polynomial corresponding to the first t primes.

Remark 2.7. The Swinnerton–Dyer polynomials $S_{\mathcal{P}}$ are typical building blocks for polynomials which are hard to factor (naïve recombination takes exponential time). They are defined inductively:

$$\begin{cases} S_{\emptyset}(X) := X \\ S_{\mathcal{P}\cup\{n\}}(X) := \mathrm{Res}_Y((X+Y)^2 - n, S_{\mathcal{P}}(Y)) & \text{if } \mathcal{P} \text{ is a list of integers.} \end{cases}$$

The polynomial $S_{\mathcal{P}}$ has degree $2^{|\mathcal{P}|}$, at least $2^{|\mathcal{P}|-1}$ modular factors, and is irreducible over $\mathbb{Q}$ provided the elements of $\mathcal{P}$ are multiplicatively independent.

Our implementation could be improved in numerous ways: in particular, polynomial arithmetic in PARI does not support the FFT or subquadratic division (integer arithmetic does), and we use an all-integral LLL for lack of a stable floating point or divide and conquer implementation in huge dimensions, as in Koy and Schnorr (2001a,b).

But it is already plain that van Hoeij's method is a major breakthrough: in many of the cases previously considered to be very difficult, Hensel lifting now dominates the running times. For instance, the polynomial P_8 was impossible to factor three years ago. Abbott et al. (2000) eventually managed to prove its irreducibility using sophisticated implementations and huge tables to prune the recombination tree in Zassenhaus algorithm (recombination took of the order of one hour of CPU time).

2.5.3. Number of LLL reductions

The following example is typical of the collapse in entry size expected from Strategy B, and always observed in practice. Let

$$P := S_{\mathcal{P}}(X) S_{\mathcal{Q}}(X) S_{\mathcal{R}}(X+1),$$

with

$$\mathcal{P} = \{2, 3, 5, 7, 11, 13, 17, 19, 23\},$$

$$\mathcal{Q} = \{2, 5, 7, 11, 13, 17, 19, -2, -3\},$$

$$\mathcal{R} = \{2, 3, 7, -11, 13, 17, 5, -7\}$$

be a "random" product of perturbed Swinnerton–Dyer polynomials. It has degree $2^9 + 2^9 + 2^8 = 1280$, at least $2^8 + 2^8 + 2^7 = 640$ modular factors, and 3 true factors of degree 2^9, 2^9 and 2^8. Using Beauzamy's bound, we can choose $p^a = 223^{543}$ throughout.

s	Initial r	δ	Successive b	# LLL runs	Time
1	640	42	543, 501, 3	2	7 266
2	634	41	543, 502, 4	2	6 155
3	604	39	543, ..., 387, 5	5	28 879
4	522	34	543, ..., 271, 6	9	45 949
5	380	25	543, ..., 218, 7	14	23 976
6	215	14	543, ..., 363, 9	13	2 094
7	93	7	543, ..., 480, 10	10	75
8	29	5	543, ..., 533, 11	3	1
9	8	5	543, 538, 12	2	0
10	4	5	543, 13	2	0

In the "successive b" column, $b_0, \ldots, b$ denotes the uneventful sequence $b_0, b_0 - \delta, b_0 - 2\delta, \ldots, b$. Roughly 31 h 54 min were needed to recover the correct factorization, among which all but 11 minutes were spent in the LLL reductions depicted above. There were 62 LLL runs in total, only 11 of which took more than 1 h; the two most expensive runs took place when $s = 3$ for about 3 h each.

3. A variant over number fields

In this section, a finite extension $\mathbb{K}$ of $\mathbb{Q}$ (a number field) is given and we let $d = [\mathbb{K} : \mathbb{Q}]$ and $\mathcal{O}_\mathbb{K}$ be its ring of integers. For basic number fields algorithms (e.g. decomposition of primes, "polynomial reduction" algorithms, ideal arithmetic), the reader is referred to Cohen (1993), Pohst (1993) and Pohst and Zassenhaus (1989). For a fractional ideal $x \subset K$, Nx denotes the absolute norm of x. By abuse of notation, we write Nx for $N(x)$ if $x \in K$.

3.1. Representation of elements in $\mathbb{K}$

By the primitive element theorem, we can write $\mathbb{K} = \mathbb{Q}(\alpha)$, where $h(\alpha) = 0$ with $h \in \mathbb{Z}[Y]$, of degree d. We choose $\alpha \in \mathcal{O}_\mathbb{K}$, i.e. h monic, so that $\mathbb{Z}[\alpha]$ is a submodule of $\mathcal{O}_\mathbb{K}$ of finite index. To describe the algorithm at the right level of generality, we fix henceforth an order $\mathcal{O}$ such that $\mathbb{Z}[\alpha] \subset \mathcal{O} \subset \mathcal{O}_\mathbb{K}$, a basis (ω_i) of $\mathcal{O}$ and a denominator $f \geq 1$ such that any $z \in \mathcal{O}_\mathbb{K}$ can be represented as

$$z = \frac{1}{f} \sum_{i=1}^{d} z_i \omega_i, \qquad \text{where } z_i \in \mathbb{Z}.$$

In particular, we do not insist that $\mathcal{O}$ be maximal ($= \mathcal{O}_\mathbb{K}$) nor f minimal (=the exponent of the additive group $\mathcal{O}_\mathbb{K}/\mathcal{O}$), in order to avoid having to compute an integral basis[3] for $\mathbb{K}$.

The precise choices depend on the given field and how much initialization time we allow. To keep initializations to a minimum, we can take $\mathcal{O} = \mathbb{Z}[\alpha]$, $\omega_i = \alpha^{i-1}$, and

[3] While well understood (it amounts to factoring h over $\mathbb{Q}_p$ for all primes p whose square divides $\mathrm{disc}(h)$), this process remains time-consuming and involves factoring the discriminant of h. See Buchmann and Lenstra Jr. (1994) and Ford et al. (2002).

compute a simple multiple f of the index. For instance we can take for f any multiple of the largest integer f_0 such that f_0^2 divides $\mathrm{disc}(h)$, e.g. $\mathrm{disc}(h)$ itself after weeding out some prime divisors found through partial factorization (see Allombert, in press for a less naïve approach using reduced discriminants).

On the other hand, we get smaller bounds, and hence better running times in our factorization algorithms, if $\mathcal{O}$ is as large as possible and f is somewhat minimal. Optimally, we can afford to compute $\mathcal{O}_{\mathbb{K}}$ and take $\mathcal{O} = \mathcal{O}_{\mathbb{K}}$, and hence $f = 1$. Also, we will see that the basis (ω_i) should be LLL-reduced for the quadratic form T_2, to be defined in Section 3.3. We shall not dwell further on this particular optimization problem, and take $(\mathcal{O}, f, (\omega_i))$ for granted.

3.2. Factorization in $\mathbb{K}[X]$

Let P be a polynomial with coefficients in $\mathbb{K}$. For simplicity, we assume that P is monic and has coefficients in $\mathbb{Z}[\alpha]$, which is achieved by a change of variables. We shall further assume that P has no square factors, which is achieved by square free factorization.

As any other factoring algorithm over $\mathbb{Q}$, van Hoeij's method can be used to factor P over $\mathbb{K}$, using Trager's method: we factor the norm

$$\mathcal{P}_\lambda := \mathrm{Res}_Y(P(X - \lambda Y), h(Y)) \in \mathbb{Q}[X],$$

where the coefficients of P are lifted from $\mathbb{K} := \mathbb{Q}[Y]/(h)$ to $\mathbb{Q}[Y]$ and λ is a small rational integer chosen so that $\mathcal{P}_\lambda$ is square free. Each $\mathbb{Q}$-factor G of $\mathcal{P}_\lambda$ is divisible by a unique $\mathbb{K}$-factor of P, which is extracted as $\gcd(G(X + \lambda Y), P)$ over $\mathbb{K}[X]$. Assuming efficient modular implementations for the initial resultants and final gcds, the main bottleneck in this reduction to the absolute case is the recombination phase during the factorization of $\mathcal{P}_\lambda$. van Hoeij's algorithm is very suitable for this kind of polynomial, and Galois resolvents in general, since it is not too sensitive to the size of coefficients, besides the Hensel lifting phase, and is able to cope with the huge number of modular factors which are often intrinsic to the method.

We shall describe an application of van Hoeij's ideas to the direct factorization in $\mathbb{K}[X]$, which is in general superior to the norm approach since the number of modular factors over $\mathbb{K}$ is smaller than that over $\mathbb{Q}$, possibly by a factor of d, without increasing too much the cost of the other steps. We follow roughly the approach of Roblot (in press), himself following Lenstra (1982), with various improvements. (Namely, using an arbitrary order instead of $\mathcal{O}_{\mathbb{K}}$, faster reconstruction through better bounds and reduction heuristics, and van Hoeij's knapsack.)

3.3. L_2 bounds

It is classical to measure the size of an element $x \in \mathcal{O}_{\mathbb{K}}$ in terms of the quadratic form $T_2(x) := \sum_\sigma |x^\sigma|^2$, where σ runs through the d embeddings of $\mathbb{K}$ into $\mathbb{C}$ and $x^\sigma := \sigma(x)$.

We first recall standard upper bounds on the coefficients of a monic factor $G = X^k + \sum_{i<k} g_i X^i$ of P in $\mathbb{K}[X]$. We derive a uniform bound on the coefficients of G by

Lemma 3.1. *Let $G = \sum_{i\le k} g_i X^i$ be a monic divisor of P as above. Then all the g_i are integral and we have*

$$T_2(g_i) \le \sum_\sigma B(i, P^\sigma)^2,$$

where $B(i, Q)$ is any bound for the modulus of the ith coefficient of a factor of $Q \in \mathbb{C}[X]$.

Proof. Let $P = GH$ in $K[X]$; then

$$\mathrm{cont}(P) = \mathrm{cont}(G)\mathrm{cont}(H),$$

where $\mathrm{cont}(A)$ denotes the fractional ideal generated by the coefficients of $A \in \mathbb{K}[X]$. Since $P \in \mathcal{O}_\mathbb{K}[X]$ is monic, we have $\mathrm{cont}(P) = \mathcal{O}_\mathbb{K}$. Since G is monic, so is H, and their contents both contain $\mathcal{O}_\mathbb{K}$. It follows that $\mathrm{cont}(G) = \mathrm{cont}(H) = \mathcal{O}_\mathbb{K}$, and hence all g_i belong to $\mathcal{O}_\mathbb{K}$. The T_2 bound is straightforward since, for each embedding $\sigma\colon \mathbb{K} \to \mathbb{C}$, the polynomial G^σ divides P^σ in $\mathbb{C}[X]$. □

We take for $B(i, Q)$ the minimum of Beauzamy's and Mignotte's bounds (cf. Beauzamy, 1992; Mignotte, 1974), computed using numerical approximations to evaluate the embeddings σ. For later reference, we also bound the Newton sums of true factors:

Lemma 3.2. *Let G be as above. Then, for all integer $k \ge 0$, the Newton sum $S_k(G)$ is in $\mathcal{O}_\mathbb{K}$. For any embedding σ, we have*

$$T_2(S_k(G)) \le \deg(P)^2 \sum_\sigma B_{\mathrm{root}}^{2k}(P^\sigma)$$

where $B_{\mathrm{root}}(Q)$ is any bound for the modulus of the complex roots of Q.

Proof. From Lemma 3.1, $G \in \mathcal{O}_\mathbb{K}[X]$. Since G is monic, the Newton sums are integral combinations of the coefficients of G, and hence integral. The bound on $T_2(S_k(G))$ is obvious by summation. □

In practical computations, it is more convenient to use our specified basis (ω_i). For $x = \sum x_i\omega_i \in \mathbb{K}$, we let $|x|^2 := \sum x_i^2$. This new quadratic form is easily related to T_2:

Lemma 3.3. *Let $M = (m_{ij}) \in M_d(\mathbb{Q})$ be the matrix such that $(\omega_i) = (\alpha^{i-1})M$, and $V = ((\alpha^\sigma)^{j-1})_{\sigma, 1\le j\le d}$ be the Vandermonde matrix associated with the complex roots (α^σ) of h; then*

$$|x|^2 \le C_{T_2} T_2(x), \qquad \text{with } C_{T_2} = \|M^{-1}V^{-1}\|_2^2,$$

where $\|(a_{i,j})\|_2 := (\sum_{i,j} |a_{i,j}|^2)^{1/2}$.

Proof. Let $x = (\omega_j)^t(x_j) = (\alpha^{j-1})M^t(x_j)$, $(x_j) \in \mathbb{Q}^d$. Writing the d different embeddings of this equation in $\mathbb{C}$, we obtain

$${}^t(x^\sigma) = V M\ {}^t(x_j), \qquad \text{hence } |x|^2 \le C_{T_2} T_2(x)$$

by Cauchy–Schwarz. □

To use Lemmas 3.1 and 3.3 as stated, it is crucial to know the roots of h with guaranteed error bounds, such as with Gourdon–Schönhage's root-finding algorithm (Gourdon, 1993).

Note that $M^{-1} \in M_d(\mathbb{Q})$ is known exactly and V^{-1} is stably evaluated to arbitrary precision from the Lagrange interpolation formula, since its ith line gives the coefficients of $E_i \in \mathbb{C}[X]$ such that $E_i(\alpha^{\sigma_j}) = \delta_{ij}$ for all j. It is possible to bound

$$\|M^{-1}V^{-1}\|_2 \leq \|M^{-1}\|_2 \|V^{-1}\|_2,$$

which requires less accuracy for a reliable evaluation, but we obtain a worse bound for a negligible computational saving. We shall explain in Remark 3.13 the practical reasons behind the choice of $|\cdot|^2$ over T_2.

Note that C_{T_2} tends to be much smaller if $((\omega_j), T_2)$ is LLL-reduced, since the L_2 norm of VM is then provably smallest possible, up to a multiplicative constant depending only on the quality ratio and the dimension d.

Corollary 3.4. *Let $\|P\|_2^2 := \sum_\sigma \|P^\sigma\|_2^2$, and assume that (w_i) is LLL-reduced for T_2. With notations as in Lemmas 3.1 and 3.3, we have*

$$T_2(g_i) = O(4^{\deg(P)}\|P\|_2^2),$$
$$C_{T_2} = O(1)^{d^2}.$$

Proof. The first estimate follows from Mignotte's bound, and the second from $VM = (w_j^\sigma)$, the formula for the matrix inverse, and the Hadamard bound, which bounds the square of a cofactor by

$$\prod_j T_2(w_j) \ll O(1)^{d(d-1)} \det(VM)^2$$

for an LLL-reduced basis. □

As usual for estimates using the LLL worst case bound, the second one is quite pessimistic: in the examples from Section 3.11, where d gets as large as 50, we always obtained $C_{T_2} < 1$.

3.4. Modular factorization

Let $\mathfrak{p}$ be a prime ideal such that $P \bmod \mathfrak{p}$ is square free, and such that $\mathfrak{p} \nmid \operatorname{disc}(h)$, so that the $\mathfrak{p}$-adic completion $\mathbb{K}_\mathfrak{p}$ of $\mathbb{K}$ is unramified. For efficiency it is worth trying several $\mathfrak{p}$ such that $P \bmod \mathfrak{p}$ has few factors in $\mathcal{O}_\mathbb{K}/\mathfrak{p}$. When choosing $\mathfrak{p}$, note that taking $\mathfrak{p}$ of small residual degree improves factorization mod $\mathfrak{p}$, Hensel lift and computation of Newton sums, while a large residual degree means easier reconstruction of algebraic numbers from $\mathfrak{p}$-adic factors; see Section 3.7.

Remark 3.5. Chebotarëv's density theorem implies that the density of primes p such that there exist $\mathfrak{p} \mid p$ of residual degree 1 is larger than $1/d$, with equality if and only if $\mathbb{K}/\mathbb{Q}$ is Galois, see Neukirch (1986, Corollary 6.5). It is not known unconditionally whether the first such p is small (assuming GRH, it is $O(\log^2 |\operatorname{disc}(\mathbb{K})|)$), but in practice it is easy to find one. On the other hand, there may not exist any prime ideal of large residual degree; for instance if $\mathbb{K}$ is a compositum of t quadratic fields, whose degree 2^t can be made arbitrarily large, the residual degree is either 1 or 2.

We lift the factorization over $\mathbb{K}_{\mathfrak{p}}$. The monic irreducible factors are computed modulo $\mathfrak{p}^a$, where a is large enough to enable reconstructing a genuine factor from its modular approximation, given bounds on the size of its coefficients. For this reconstruction, we shall improve on a method of Lenstra (1982).

3.5. Preliminaries on lattices

Definition 3.6. Let E be a Euclidean space, $\Lambda \subset E$ a lattice with given basis $\mathfrak{B} = (b_i)$.

(1) We define the (open, centred) fundamental domain associated with $\mathfrak{B}$ by

$$\mathfrak{F} = \mathfrak{F}(\mathfrak{B}) := \left\{ \sum \lambda_i b_i,\ (\lambda_i) \in \mathbb{R}^{\mathfrak{B}},\ |\lambda_i| < 1/2 \right\}.$$

(2) Let $B(0, r)$ be the open ball of radius r, centred at 0. We denote by

$$r_{\max} = r_{\max}(\mathfrak{B}) := \sup\{r \in \mathbb{R}^+,\ B(0, r) \subset \mathfrak{F}\}$$

the radius of the largest ball inscribed in the closure of $\mathfrak{F}$.

We solve the reconstruction problem using the following:

Lemma 3.7. *Let E, Λ, $\mathfrak{B}$, $\mathfrak{F}$, $r_{\max}$ be as in 3.6, and $|\ |^2$ the Euclidean norm. If $x \in E$, there is at most one $y \in E$ such that*

$$x \equiv y \pmod{\Lambda} \qquad \text{and} \qquad |y| < r_{\max}.$$

If it exists, y is the unique element in $\mathfrak{F}$ congruent to x modulo Λ. In terms of coordinates (on a fixed arbitrary basis), let M be the matrix giving the (b_i); then y is given by

$$y = x \bmod M := x - M\lfloor M^{-1}x \rceil. \tag{5}$$

As usual, $\lfloor x \rceil := \lfloor x + 1/2 \rfloor$ is the operator rounding to nearest integer and is to be applied coordinatewise.

Proof. If $|y| < r_{\max}$, then by definition $y \in \mathfrak{F}$. If $y' = y + \lambda$ is another solution, with $\lambda \in \Lambda$, then $|\lambda| \le |y| + |y'| < 2r_{\max}$, and hence $\lambda \in 2\mathfrak{F} \cap \Lambda = \{0\}$. The other assertions are obvious. □

The radius $r_{\max}$ is computed in the following way:

Lemma 3.8. *Let (b_i^*) be the orthogonalized vectors corresponding to (b_i), and let R be the (upper triangular) Gram–Schmidt base change matrix: $(b_i) = (b_i^*)R$. Let $(t_{i,j}) := R^{-1}$; then*

$$r_{\max} = \min_i \frac{1}{2T_i}, \qquad \text{where} \qquad T_i := \left(\sum_j t_{i,j}^2 / |b_j^*|^2 \right)^{1/2}.$$

Proof. Let $x = \sum_i \lambda_i b_i = \sum_i \lambda_i^* b_i^*$, with coordinates $\lambda_i, \lambda_i^* \in \mathbb{R}$. By definition of R, we have the relation $\lambda_i = \sum_j t_{i,j}\lambda_j^*$ for all i. Provided $|x|T_i < 1/2$, Cauchy–Schwarz yields

$$\lambda_i^2 = \left(\sum_j t_{i,j}\lambda_j^* \right)^2 \le \sum_j (\lambda_j^*)^2 |b_j^*|^2 \sum_j t_{i,j}^2 / |b_j^*|^2 = |x|^2 T_i^2 < \tfrac{1}{4}. \tag{6}$$

Conversely, let i be an index such that T_i is maximal and

$$x := \frac{1}{2T_i^2} \sum_j \frac{t_{i,j}}{|b_j^*|^2} b_j^*,$$

defining implicitly the (λ_j) and (λ_j^*). Then all inequalities in Eq. (6) become equalities and $|x| = 1/(2T_i) = \min_j 1/(2T_j)$ implies that x is on the boundary of $\mathfrak{F}$ by the previous argument (since $|\lambda_j| \leq 1/2$ for all j, and $\lambda_i = 1/2$). □

For a fixed Λ, a convenient way of maximizing $r_{\max}$ is to use an LLL-reduced basis, which is close to orthogonal, with $|b_i^*|$ not much smaller than $|b_1^*|$. From a theoretical point of view:

Lemma 3.9 (Lenstra, 1982). *We have*

$$r_{\max}(\mathfrak{B}) \geq r_0 := \frac{1}{2} \min_i |b_i| \times \prod_j \frac{|b_j^*|}{|b_j|}.$$

Proof. The product $\prod |b_i^*|$ is the volume of the fundamental domain $\mathfrak{F}$, and hence is independent of the ordering of the (b_i). Let $x = \sum_i \lambda_i b_i = \sum_i \lambda_i^* b_i^*$; by symmetry, it is enough to prove that

$$|x| < r_0 \Longrightarrow |\lambda_n| < 1/2.$$

Since the Gram–Schmidt base change matrix R is triangular with identity diagonal, we have $\lambda_n = \lambda_n^*$. Hence $|x| = \lambda_n |b_n^*| + |u|$, where u is the orthogonal projection of x on $\langle b_1, \ldots, b_{n-1} \rangle$. Finally, $|x| < r_0$ implies

$$|\lambda_n| < r_0/|b_n^*| \leq \tfrac{1}{2} \prod_{i \neq n} (|b_i^*|/|b_i|) \leq \tfrac{1}{2}$$

since $\min_i |b_i| \leq |b_n|$ and $|b_i^*| \leq |b_i|$ for all i. □

Assume that Λ is LLL-reduced with quality ratio α; let $\gamma := (\alpha - 1/4)^{-1} \geq 4/3$ and $d := \dim E$. Then, from the properties of an LLL-reduced basis (see e.g. Cohen, 1993, Theorem 2.6.2), Lenstra obtains that

$$r_{\max}(\mathfrak{B}) \geq \frac{\min_i |b_i|}{2\gamma^{d(d-1)/4}}.$$

A direct study using Lemma 3.8 yields the following improvement:

Proposition 3.10. *Assume (b_i) is LLL-reduced with quality ratio $1/4 < \alpha \leq 1$; let $\gamma := (\alpha - 1/4)^{-1} \geq 4/3$ and $d := \dim E$. Then*

$$r_{\max}(\mathfrak{B}) \geq \frac{|b_1|}{2(3\sqrt{\gamma}/2)^{d-1}}.$$

Proof. The matrix R is upper triangular, with identity diagonal; hence $t_{i,i} = 1$, $1 \leq i \leq d$. Since (b_i) is size-reduced, the off-diagonal entries of R are bounded by $1/2$. By induction (or by a straightforward estimation of the cofactor of $r_{i+k,i}$), we obtain $|t_{i,i+k}| \leq (1/2)(3/2)^{k-1}$ for $1 \leq i \leq d$, $1 \leq k \leq d - i$. From the properties of

LLL-reduced bases, we have $|b_1^*|^2 \le \gamma^{i-1}|b_i^*|^2$, for $i \ge 1$ (Cohen, 1993, proof of Theorem 2.6.2). It follows that

$$\begin{aligned}|b_1^*|^2 \sum_{j=1}^{d} t_{i,j}^2/|b_j^*|^2 &\le \sum_{k=0}^{d-i} t_{i,i+k}^2 \gamma^{i+k-1}\\ &\le \gamma^{i-1} + (\gamma^i/4)\sum_{k=1}^{d-i}(9\gamma/4)^{k-1}\\ &= \gamma^{i-1} + \gamma^i\frac{(9\gamma/4)^{d-i}-1}{9\gamma-4} < (9\gamma/4)^{d-1}\end{aligned}$$

since $\gamma \ge 4/3 > 1/2$ implies that $1/(9\gamma-4) < 1/\gamma$. The result follows since $b_1^* = b_1$. □

The actual bound given by the proof is slightly sharper; we use this simpler form since we shall not need its precise value. Also, as d tends to infinity, the tighter bound does not improve significantly on $\log r_{\max}$, which is the significant parameter here.

Remark 3.11. Note that $r_{\max} \le \frac{1}{2}\min_i |b_i^*| \le \frac{1}{2}|b_1|$, with equality when (b_i) is orthogonal and b_1 is a shortest vector in the basis. For an LLL-reduced basis, Proposition 3.10 proves that the right-hand side is close to $r_{\max}$, up to a multiplicative constant depending only on the dimension and the LLL quality ratio.

3.6. Reconstruction from $\mathfrak{p}$-adic approximation

Assume we want to reconstruct algebraic integers $x \in \mathcal{O}_{\mathbb{K}}$ satisfying $T_2(x) < C$. We could use the following a priori bound:

Lemma 3.12. *Let (b_i) be a basis of $\Lambda = \mathfrak{p}^a$, LLL-reduced for T_2 with quality ratio α, and let $\gamma := 1/(\alpha - 1/4)$, $d := [\mathbb{K} : \mathbb{Q}]$. Let $C > 0$ and $x \in \mathcal{O}_{\mathbb{K}}$ be such that $T_2(x) < C$; then we can apply Lemma 3.7 to reconstruct x uniquely from $x \pmod{\mathfrak{p}^a}$, provided*

$$N\mathfrak{p}^a \ge (2\sqrt{C/d}\cdot(3\sqrt{\gamma}/2)^{d-1})^d.$$

Proof. We apply Lemma 3.7 to $\Lambda = (\mathfrak{p}^a, T_2)$. Since $b_1 \in \mathfrak{p}^a - \{0\}$, we have

$$T_2(b_1) \ge d(Nb_1)^{2/d} \ge d(N\mathfrak{p}^a)^{2/d},$$

where the first inequality follows from the arithmetic geometric means and the second from the fact that $N\mathfrak{p}^a \mid Nb_1$, with $Nb_1 \ne 0$. From Proposition 3.10, it follows that

$$r_{\max} \ge \frac{\sqrt{d}(N\mathfrak{p}^a)^{1/d}}{2(3\sqrt{\gamma}/2)^{d-1}},$$

and we can apply Lemma 3.7 as soon as $r_{\max} > \sqrt{C}$. □

This bound, in particular the term $(3\sqrt{\gamma}/2)^{d-1}$, is rather pessimistic, and its main purpose is to show that a is polynomial in $\log C$ and d. In fact, it is $O(d^2 + d\log C)$, whereas Lenstra and Roblot used a bound in $O(d^3 + d\log C)$.

It follows from the preceding discussions, and in fact from Lenstra's original argument, that the LLL reduction of $\mathfrak{p}^a$ is achieved in polynomial time with respect to d and $\log C$.

Nevertheless, it is very expensive in practice: we shall see that it quickly dominates the running time when d gets large. Hence, in order to reconstruct $x \in \mathcal{O}_{\mathbb{K}}$ such that $|x|^2 < C_{T^2}C$, we rather proceed as follows:

(1) Start from a heuristic value of a; say, the one from Lemma 3.12 barring the $(3\sqrt{\gamma}/2)^{d-1}$ term.
(2) Find an LLL-reduced basis $\mathfrak{B}$ of $\Lambda = (\mathfrak{p}^a, |\,|^2)$.
(3) Compute $r_{\max}(\mathfrak{B})$ exactly, using Lemma 3.8. Note that R and the squared lengths of the b_i^* are computed during the LLL algorithm; these computations are exact since Λ has exact entries. The matrix R is upper triangular, with ones on the diagonal, and hence easily inverted. This inversion and the weighted L^2 norm computations are stable, yielding a guaranteed sharp lower bound for $r_{\max}$ with $O(d^3)$ low accuracy floating point computations.
(4) If $r_{\max}^2 > C_{T_2}C$ we are done; otherwise double a and restart from Step (2).

We shall apply these results for *increasing* values of C. We use a given $\mathfrak{p}$-adic precision and given reduced basis as long as $r_{\max}^2 \geq C_{T_2}C$. In practice, these improvements have a significant impact when factoring small polynomials over large fields, but are negligible when $\log C \gg d$.

Remark 3.13. We have chosen $|\cdot|^2$ instead of the more natural T_2 to be able to use an integral LLL algorithm in Step (2). Choosing T_2 yields smaller bounds (only slightly so, provided (ω_i) is LLL-reduced), but would force us to compute embeddings to a huge accuracy to avoid stability problems during the LLL reduction. This idea is due to Fieker and Friedrichs (2000).

Remark 3.14. A convenient feature of Lenstra and Roblot's method is that rational reconstruction is given by a formula (5), once the expensive preliminary LLL reduction of $\mathfrak{p}^a$ is done. This formula is crucial to our extension of van Hoeij's method, especially the truncation of low order bits in Section 3.9.

Pohst has suggested using a smaller value of a, and enumerating short vectors in congruence classes modulo $\mathfrak{p}^a$ instead. (This variant is implemented in the KANT system (Daberkow et al., 1997).) In practice, Pohst's bound is the one from Lemma 3.12, barring the γ term coming from the uncertainty on the LLL output:

$$N\mathfrak{p}^a \geq (2\sqrt{C/d})^d,$$

so it probably provides at most a minor improvement on the method suggested above. Also, we do not see how to adapt van Hoeij's method in this framework.

3.7. Practical improvements to the reconstruction LLL

The drawback of relative factorization methods is that reconstruction of algebraic numbers from $\mathfrak{p}$-adic approximations is costly when d is large, since it requires the LLL reduction of a lattice of dimension d and huge entries. If many factorizations over the same ground field are required, this reduction need only be performed once, provided the coefficient bounds are of the same order of magnitude, and polynomials to be factored remain square free modulo the chosen prime. The following discussions assume that the reconstruction LLL is not part of the precomputations.

To reconstruct x with $|x| < C$, Lemma 3.12 requires $\log N\mathfrak{p}^a \approx d^2 + d\log C$. An HNF matrix for $\mathfrak{p}^a$ has entries bounded by $N\mathfrak{p}^{a/f}$, where f is the residual degree of $\mathfrak{p}$. For entries of bitsize B, the original LLL runs in at most $O(Bd^4)$ operations on numbers of bitsize $O(Bd)$; see Lenstra et al. (1982, Proposition 1.26). For a fixed polynomial P, we have $\log C = O_P(d^2)$ by Corollary 3.4 and hence $B = O_P(d^3/f)$; for a prime ideal $\mathfrak{p}$ of degree 1, this yields an impressive upper bound of $O_P(d^{11+\varepsilon})$ bit operations for every $\varepsilon > 0$, using asymptotically fast arithmetic. Assuming $B = O_P(d^2)$, which is closer to the observed behaviour, this would go down to $O_P(d^{9+\varepsilon})$. Although polynomial in d, this gets formidable as d increases.

When $\deg(P)$ is small compared to d, Trager's algorithm is expected to be faster than relative methods, even though the latter incur no recombination time while the former may have a large number of modular factors and involves taking gcds over a large number field. Even for tough Galois resolvents, this single LLL reduction may well dominate the running times (Section 3.11.5). We use the following specific reduction strategy:

(1) If the field degree d is small, use prime ideals $\mathfrak{p}$ of degree 1 for faster Hensel lift, Newton sums, and reconstruction of algebraic numbers (once the initial LLL reduction is done). Otherwise look for higher residual degree, taking care not to cripple factorization modulo $\mathfrak{p}$. If $\mathfrak{p}$ is far from inert, the matrix M of $\mathfrak{p}^a$ in HNF is badly skewed.
(2) Use Peter Montgomery's heuristic partial LLL algorithm (unpublished, `lllintpartial` in PARI) on M. The resulting matrix M' is *partially reduced* (two distinct columns c_1, c_2 satisfy $|c_1 \pm c_2| \geq |c_1|$) and has expected sup norm around $N\mathfrak{p}^{a/d}$. In general, it is far from being LLL-reduced.
(3) Apply a floating point LLL reduction to M' following Schnorr–Euchner strategy (Schnorr and Euchner, 1994), with Householder orthogonalization.

The partial reduction in Step (2) yields important practical speed-ups: for a prime ideal of residual degree f, the expected bitsize of the matrix entries is divided by d/f. In practice, more than 80% of the reduction time is spent in the last step.

3.8. The relative knapsack

To adapt van Hoeij's algorithm, we first introduce the adapted knapsack problem, and postpone the relative truncation to the next section. We begin by fixing a lift $\mathcal{O}_{\mathbb{K}_\mathfrak{p}}/\mathfrak{p}^a \to \mathbb{Z}^d$. The ideal $\mathfrak{p}^a \cap \mathcal{O}$ is represented by a $\mathbb{Z}$-basis: $(\omega_i)M$, where M is an integral $d \times d$ matrix. We define $\mathrm{lift}(x) := x \bmod \mathfrak{p}^a$, where the latter is defined by taking any representative $y = \sum y_i\omega_i$ of x in $\mathcal{O}$, then reducing (y_i) modulo M, as in Lemma 3.7. This coincides with the centred representative used over $\mathbb{Q}$ for $p \neq 2$. The basic knapsack lattice is given by

$$\begin{pmatrix} C\mathrm{Id}_n & 0 \\ S & Q \end{pmatrix}$$

where

- $C \geq 1$ is a suitable integer constant (see below).

- $S = \mathrm{lift}(f S_{\mathbb{K}_\mathfrak{p}})$ is a $(Nd) \times n$ matrix, where

$$S_{\mathbb{K}_\mathfrak{p}} := \begin{pmatrix} S_{i_1} G_1 & \dots & S_{i_1} G_n \\ \vdots & \vdots & \vdots \\ S_{i_N} G_1 & \dots & S_{i_N} G_n \end{pmatrix},$$

- Q is a $(Nd) \times (Nd)$ block diagonal matrix, with blocks equal to M on the diagonal.

The Newton sums are computed from the Newton formulae as in the absolute case, and are bounded using Lemma 3.2. To any true factor of P there correspond $u \in \{0,1\}^n$ and $v \in \mathbb{Z}^{Nd}$ such that the image of $\binom{u}{v}$ has squared L_2 norm bounded by

$$C^2 n + \|Su + Qv\|_2^2$$

and we can bound $\|Su + Qv\|_2 \le B_{\mathrm{trace}}$ using Lemmas 3.2 and 3.3, with

$$B_{\mathrm{trace}}^2 := f^2 C_{T_2} \deg(P)^2 \sum_{k=1}^{N} \sum_{\sigma} B_{\mathrm{root}}^{2i_k}(P^\sigma). \tag{7}$$

The constant C is chosen so that $C^2 n \approx B_{\mathrm{trace}}^2$. It is not necessary at this point that M be LLL-reduced, nor that we use the lift specified above, although both conditions certainly speed up the reduction.

3.9. Truncation

For $t > 1$ any integer we now define the truncation $\mathcal{T}_{\mathbb{K}}^{a,t} : \mathcal{O}_{\mathbb{K}_\mathfrak{p}}/\mathfrak{p}^a \to \mathbb{Z}^d$ by $\mathcal{T}_{\mathbb{K}}^{a,t}(x) := \lfloor \mathrm{lift}(x)/t \rceil$. Over $\mathbb{Z}$, we could have used the operator $\mathcal{T}_{\mathbb{Q}}^{a,p^b}$ instead of $T^{a,b}$, since they are equal modulo p^{a-b} and yield equally small entries.

To follow exactly van Hoeij's argument, we need truncated elements to belong to $\mathcal{O}_{\mathbb{K}}$. In general, we do not control what happens at $\mathfrak{q} \mid p$, $\mathfrak{q} \neq \mathfrak{p}$, and hence we cannot divide out by powers of p and preserve integrality, so we do not insist that t be a power of p. On the other hand, if p is inert, then van Hoeij's idea carries through and, choosing $t = p^b$, we obtain better bounds. Of course, such a p may not exist.

Let $t > 1$ be any integer; we write $S = S_0 + tS_1$, where $\|S_0\|_\infty \le t/2$ (so that $S_1 = \mathcal{T}^{a,t}(f S_{\mathbb{K}_\mathfrak{p}})$ applied blockwise), and likewise $Q = Q_0 + tQ_1$; hence

$$\|S_0\|_2 \le \sqrt{Ndn}\frac{t}{2} \qquad \text{and} \qquad \|Q_0\|_2 \le \sqrt{Nd}\frac{t}{2}.$$

From the previous section, there exist $u \in \{0,1\}^n$, $v \in \mathbb{Z}^{Nd}$ such that

$$\|Su + Qv\|_2 \le B_{\mathrm{trace}}.$$

Hence

$$\begin{aligned} \|S_1 u + Q_1 v\|_2 &\le B_{\mathrm{trace}}/t + \|S_0 u + Q_0 v\|_2/t \\ &\le B_{\mathrm{trace}}/t + \frac{\sqrt{Ndn}}{2}\|u\|_2 + \frac{\sqrt{Nd}}{2}\|v\|_2. \end{aligned}$$

In van Hoeij's original argument, we can assume that Q is divisible by t, and hence $Q_0 = 0$ and the last term drops out, so it is not important to control v. Here, we have to make two further assumptions:

- The matrix M associated with $\mathfrak{p}^a$ is such that we can apply Lemma 3.7, which is the case if a is so large that $r_{\max} \geq B_{\text{trace}}$. From this we deduce that $v = -\lfloor Q^{-1}Su \rceil$.
- The specified $\mathfrak{p}$-adic lift is chosen for S, so that $\|Q^{-1}S\|_\infty \leq 1/2$.

For a vector $x \in \mathbb{R}^d$, we have $\lfloor x \rceil = x + \varepsilon$, where $\|\varepsilon\|_\infty \leq 1/2$; hence

$$\|\lfloor x \rceil\|_2 \leq \|x\|_2 + \sqrt{d}/2,$$

from which we gather

$$\|v\|_2 \leq \|Q^{-1}Su\|_2 + \frac{\sqrt{d}}{2} \leq \frac{\sqrt{Ndn}}{2}\|u\|_2 + \frac{\sqrt{d}}{2}.$$

Finally, using $\|u\|_2 \leq \sqrt{n}$, we obtain

$$\|S_1 u + Q_1 v\|_2 \leq B_{\text{high}} \coloneqq B_{\text{trace}}/t + \frac{n\sqrt{Nd}}{2}(1 + \sqrt{N}d/2) + \frac{d\sqrt{Nd}}{4}. \tag{8}$$

So our final knapsack lattice is given by

$$\begin{pmatrix} C\mathrm{Id}_n & 0 \\ S_1 & Q_1 \end{pmatrix}$$

where $C \geq 1$ is chosen so that $C^2 n \approx B_{\text{high}}^2$. We use either strategy A or B at this point, strategy B being by far the most efficient.

Remark 3.15. Note that $\|Q_0\|_2$, $\|S_0\|_2$, and $\|Q^{-1}S\|_2$ are cheaply and explicitly computed, resulting in sharper practical bounds; in fact we can do even better and bound directly $\|S_0 u\|_2$ and $\|Q^{-1}Su\|_2$, $u \in \{0,1\}^n$, via the ad hoc norm $\|\cdot\|$ from Strategy A for a negligible extra cost.

If p is inert and we choose t a power of p, then $Q_0 = 0$, in which case the bound becomes $B_{\text{trace}}/t + n\sqrt{Nd}/2$. Here we essentially recover van Hoeij's bound of $n\sqrt{N}/2$ in the case $d = 1$ ($\mathbb{K} = \mathbb{Q}$), provided we can take $t \gg B_{\text{trace}}$, i.e. provided that modular factors have been sufficiently lifted.

3.10. $d - k$ tests

The $d - k$ tests generalize to the relative situation: we precompute $S \coloneqq \mathcal{T}^{a,1}(S_k(G_i))$, $1 \leq i \leq n$, and $S_1 \coloneqq \mathcal{T}^{a,t}(S)$ where t is such that the coordinates of S_1 fit into machine integers. Now Q and Q_1 are defined as above and we check that

$$\|S_1 u - Q_1 \lfloor Q^{-1}Su \rceil\|_2 \leq B_{\text{high}}$$

(these are the high order digits of $Su \pmod{\mathfrak{p}^a}$), before trying the putative factor $\prod_i G_i^{u_i}$, with $(u_i) \in \{0,1\}^n$.

The test is cheap since multiplication by u is simply the sum of a column selection; S_1 and Q_1 are fully precomputed, with small entries by the choice of t. The matrix $Q^{-1}S$ is also precomputed, but it has rational entries of large height this time. Hence, we replace it

by a floating point approximation, so that rough approximations of $Q^{-1}Su$ can be quickly obtained, and the exact computation need only be done when $\lfloor Q^{-1}Su \rceil$ cannot be deduced with certainty, i.e. some coordinate is close to the mid-point between two integers. This is expected to happen with small probability: heuristically, assuming (wrongly) a uniform distribution, an entry x would satisfy

$$|x - \lfloor x \rceil - 1/2| < \varepsilon/2 \leq 1/2$$

with probability ε. In any case, this check is not very costly: only the dubious coordinates need to be recomputed, each of them essentially at the cost of a single division by the denominator of $Q^{-1}S$.

3.11. Examples

3.11.1. Implementation comments

The generalized Zassenhaus's algorithm uses a fixed bound derived from Lemmas 3.1 and 3.3. By Eq. (7), the bound B_{trace} increases as $(\max_\sigma B_{\text{root}}(P^\sigma))^k$ with the trace index k. Section 2.2 on the importance of sharp estimates for B_{root} is even more important in the number field case since weak bounds lead to extra, larger, reconstruction lattices. In the relative situation, we use tight bounds derived from the Graeffe variants of Gourdon (1993, Chapter 3) (which were deemed too expensive over $\mathbb{Q}$).

In our implementation, we use Trager's method if $3\deg(P) < [\mathbb{K} : \mathbb{Q}]$. Otherwise we first try a naïve recombination of all sets of modular factors with three elements or fewer; if the number of modular factors is smaller than 10, the factorization is run to completion in this way. For the knapsack, the tuning parameters

$$N = 1, \qquad \texttt{BitsPerFactor} = 0.5$$

were determined experimentally to give the best performance on average; other values of `BitsPerFactor` between 0.25 and 0.75 give analogous timings, with more variation as in the absolute setting.

From now on, (T) denotes Trager's absolute method, and (H) is the relative factorizer introduced in this paper, generalizing van Hoeij's ideas. As before, all timings are given in seconds.

3.11.2. Simple situations

We start with a simple test to show the power of the relative methods: let $\Phi_n(X)$ be the nth cyclotomic polynomial. The time needed to factor $\Phi_n(X + Y)$ over $\mathbb{K} = \mathbb{Q}[Y]/(h(Y))$ for all $n \leq 100$ and various h is as follows:

$h(Y)$	Time (H)	Time (T)
$Y^2 - 2$	1.8	7.9
$Y^3 - 3$	2.4	20.6
$\Phi_5(Y)$	3.4	45.0
$Y^5 - 5$	3.5	93.7
$\Phi_7(Y)$	5.0	215.6
$\Phi_{11}(Y)$	11.9	598.9

This is a typical behaviour for small base fields and simple polynomials, completely factored by a relative naïve recombination, whereas their norm can only be factored over $\mathbb{Q}$ by using van Hoeij's technique, with some difficulty.

3.11.3. Reconstruction trouble

Let us now consider polynomials of very small degree, in large fields. A trivial example is $(X-\alpha)(X+\alpha)$ over $\mathbb{Q}(\alpha)$ for $\alpha = 2^{1/32}$. (T) factors a polynomial of degree 64 and takes about 0.02 s altogether, without any arithmetic input concerning $\mathbb{K}$.

Switching to (H), we take $\mathcal{O} = \mathcal{O}_{\mathbb{K}}$ and eventually obtain a relative L_2 bound of 105.8; using a prime of degree 1 above 7, and the standard quality ratio $\alpha = 3/4$, Lemma 3.12 provides the pessimistic exponent $a = 145$, and the reconstruction LLL requires 20 s. The minimal exponent depends on a number of non-canonical choices; in our implementation, it is equal to $a = 85$. Even using that best possible exponent, about 0.5 s are still spent computing and LLL-reducing $\mathfrak{p}^a$. Picking the inert prime above 5, these computations become instantaneous, but the relative method remains slower (0.2 s).

3.11.4. Difficult examples

We now switch to more interesting polynomials. Here, it is already slower to compute Trager's resultant, let alone factor it: its degree is 2^{11}, with huge coefficients and at least 2^{10} modular factors.

- $S_7(X)$ over $\mathbb{Q}[Y]/(S_4(Y))$, 2^6 modular factors: (H) finds the 2^4 factors in about 42 s; respectively 30%, 30%, and 10% of the time is spent reducing the (unique) reconstruction lattice, reducing the recombination lattices, and in the Hensel lift. Since only 2^2 modular factors are needed for each true factor, naïve relative recombination is in fact faster (24 s, using $d-1$ and $d-2$ tests).
- $S_8(X)$ over $\mathbb{Q}[Y]/(S_3(Y))$, 2^7 modular factors: this is unfeasible with a naïve approach since 2^4 modular factors are now needed for each of the 2^3 true factors. (H) deals with it in 81 s, 1%, 55%, and 20% of which being spent in the reconstruction and knapsack LLL, and Hensel lift respectively; 15% was spent in the fruitless naïve recombination.

3.11.5. Increasing the base field degree

We factor a product of two perturbed Swinnerton–Dyer polynomials $(S_7S_8)(X+Y)$ over various random number fields $\mathbb{Q}[Y]/(h(Y))$ of increasing degrees, selected[4] so that we can take $\mathcal{O} = \mathcal{O}_{\mathbb{K}}$. This polynomial has at least 192 modular factors, but only 2 true factors unless the base field contains one of $\mathbb{Q}(\sqrt{2}), \ldots, \mathbb{Q}(\sqrt{19})$, which only occurs in degree 2 below.

In order to study worst case reconstruction, we force the algorithm to use a prime of residual degree $f = 1$, except for $[\mathbb{K}:\mathbb{Q}] = 50$ (see below). The following table gives the time in seconds needed for set-up (computing bounds, reducing and factoring P modulo various prime ideals, Hensel lift), computing and checking tentative factors (during the

[4] The fields are obtained by the following process: randomly generate a polynomial with sup norm less than 10; eliminate if the integral basis of the associated field is not easily computed, otherwise pipe it through a polynomial reduction algorithm.

naïve recombination[5] or at the end of the knapsack), the single reconstruction LLL, and the various knapsack LLL.

$[\mathbb{K}:\mathbb{Q}]$	Bounds	Mod $\mathfrak{p}$	Lift	Check	Rec LLL	Knap LLL	Total
1	0.7	2	8	1	0	182	195
2	3	2	8	6	0.0	223	244
3	7	2	15	8	0.0	211	245
4	9	2	21	16	0.1	286	337
5	12	2	31	15	0.4	347	415
6	14	3	40	20	0.8	299	387
7	16	3	57	25	2	356	476
8	17	3	61	41	3	348	495
9	22	3	89	39	8	393	591
10	27	4	101	45	12	449	682
11	27	4	105	51	17	476	734
12	31	4	121	60	29	555	868
13	33	5	139	73	46	602	997
14	34	4	156	86	72	688	1 149
15	38	5	175	101	99	751	1 302
20	51	6	270	170	489	1154	2 445
30	83	11	523	625	4 174	2567	9 149
50 ($f = 1$)	146	18	1152	1736	72 429	9117	90 123
50 ($f = 3$)	146	600	2289	917	44 360	9159	59 166
50 ($f = 9$)	146	6 166	3096	563	12 435	9129	32 014
50 ($f = 19$)	146	40 846	3060	375	4 707	9077	58 401

Experimentally, the dependence on d is not as bad as the worst case $O(d^{11+\varepsilon})$ from Section 3.7 could suggest: least square fitting for Cd^{α} yields $\alpha \approx 5.5$. The floating point strategy helps a lot; for instance, the largest reconstruction we have attempted with an all-integral LLL was in degree $[\mathbb{K} : \mathbb{Q}] = 12$: it required 1540 s, instead of the above 29 s. But reconstruction LLL using primes of degree $f = 1$ eventually dominates running times.

In degree 50, we give the timings associated with some prime ideals of larger residual degree f; the norm of $\mathfrak{p}^a$ was respectively $53^{1\times 11310}$, $47^{3\times 3888}$, $271^{9\times 891}$, and $227^{19\times 436}$, all satisfying $\log N\mathfrak{p}^a \approx 44\,900$. An optimum is attained when balancing the costs of factorization mod $\mathfrak{p}$ and reconstruction LLL, around $f \approx 10$ in this specific case.

The polynomials used to define the base fields were as follows:

$$
\begin{aligned}
h_1 &= y - 1,\\
h_2 &= y^2 - 19,\\
h_3 &= y^3 - y^2 + 6,\\
h_4 &= y^4 - 2y^3 - 7y^2 - 7y + 3,\\
h_5 &= y^5 - 3y^3 - 2y^2 + 8y - 10,
\end{aligned}
$$

[5] Due to the nature of the polynomial factored, this step is a waste of time. We include it as an indication of the performance of the generic algorithm: it checks $\binom{192}{1} + \binom{192}{2} + \binom{192}{3} = 1179\,808$ small factors, and a few large ones at the end of the knapsack.

$$
\begin{aligned}
h_6 &= y^6 - 5y^4 - y^3 + 10y^2 - 11y + 5,\\
h_7 &= y^7 - y^6 - 32y^5 - 100y^4 - 130y^3 - 70y^2 + 4y + 23,\\
h_8 &= y^8 - 8y^6 - y^5 + y^3 + 3y^2 - 7y + 2,\\
h_9 &= y^9 - y^8 - 42y^7 - 180y^6 - 375y^5 - 452y^4 - 327y^3 - 133y^2 - 31y - 11,\\
h_{10} &= y^{10} - 2y^9 + 5y^8 + 8y^7 + 5y^6 + 8y^5 - 9y^4 - 6y^3 + 7y^2 + 4y + 8,\\
h_{11} &= y^{11} - 5y^{10} + 3y^9 - 5y^8 + y^7 - 5y^6 + 8y^5 + 7y^4 - y^3 + 2y^2 - 6y + 9,\\
h_{12} &= y^{12} - 4y^{11} + 7y^{10} - 8y^9 + 8y^8 - 5y^7 + 8y^6 - 5y^4 - 3y^3 - 6y^2 + y - 6,\\
h_{13} &= y^{13} - 5y^{12} + 8y^{11} - 10y^{10} + 3y^9 + 7y^8 + 6y^7 - 5y^6 - 9y^5\\
&\quad + 5y^4 - 8y^3 + 8y^2 + y - 8,\\
h_{14} &= y^{14} - 3y^{13} - 4y^{12} - 8y^{11} - 8y^{10} + 3y^9 + 5y^8 + 4y^7 + 9y^5\\
&\quad + 5y^4 + 3y^3 - 2y^2 + 4y - 7,\\
h_{15} &= y^{15} - 3y^{14} - 4y^{13} - 8y^{12} - 8y^{11} + 3y^{10} + 5y^9 + 4y^8 + 9y^6\\
&\quad + 5y^5 + 3y^4 - 2y^3 + 4y^2 - 7y + 7,\\
h_{20} &= y^{20} - 3y^{19} - 4y^{18} - 8y^{17} - 8y^{16} + 3y^{15} + 5y^{14} + 4y^{13} + 9y^{11} + 5y^{10}\\
&\quad + 3y^9 - 2y^8 + 4y^7 - 7y^6 + 7y^5 - 6y^4 - 4y^3 + 4y^2 + 6y + 3,\\
h_{30} &= y^{30} + y^{28} + 3y^{27} + 2y^{26} - 8y^{25} - y^{24} - 7y^{23} - 8y^{21} - 8y^{20} + 6y^{19}\\
&\quad - 10y^{18} + 2y^{17} + 6y^{16} - 10y^{15} - 10y^{14} - 10y^{13} - 9y^{12} - 3y^{11}\\
&\quad - 8y^{10} + 6y^9 - 5y^8 - 9y^7 - y^6 + 7y^5 + 4y^4 + y^3 + y^2 - 5,\\
h_{50} &= y^{50} + 3y^{49} - 4y^{48} + 8y^{47} - 8y^{46} - 3y^{45} + 5y^{44} - 4y^{43} - 9y^{41} + 5y^{40}\\
&\quad - 3y^{39} - 2y^{38} - 4y^{37} - 7y^{36} - 7y^{35} - 6y^{34} + 4y^{33} + 4y^{32} - 6y^{31}\\
&\quad + 3y^{30} - 2y^{29} - 9y^{28} + y^{27} - 9y^{26} - 10y^{25} - y^{24} - 8y^{23} + 6y^{22}\\
&\quad + 6y^{21} + 5y^{20} + 2y^{19} + 3y^{18} - 6y^{17} - 10y^{15} - 4y^{14} - y^{13} - y^{12} - 2y^{11}\\
&\quad + 2y^{10} - 10y^9 + 8y^8 - 3y^7 - 6y^6 + 6y^5 - 8y^4 + 8y^3 + y^2 - 6y + 5.
\end{aligned}
$$

Acknowledgements

This work started out as a joint project with Guillaume Hanrot and Paul Zimmermann, who proposed reducing the lattice dimension (Strategy A), and implemented van Hoeij's algorithm using the NTL library (Shoup, 2003). These results were circulated in the technical report (Belabas et al., 2001), which unfortunately contained two distinct gaps, invalidating both the reported timings over $\mathbb{Q}$ and the proposed generalization to number fields.

I am indebted to Bill Allombert, Guillaume Hanrot, Mark van Hoeij and Paul Zimmermann for helpful discussions and comments. All timings were obtained thanks to the *Unité Mixte de Service* Médicis (http://www.medicis.polytechnique.fr/).

References

Abbott, J., Shoup, V., Zimmermann, P., 2000. Factorization in $\mathbb{Z}[x]$: the searching phase. In: Traverso, C. (Ed.), ISSAC'2000. ACM Press, pp. 1–7.

Allombert, B. An efficient algorithm for the computation of Galois automorphisms, Math. Comp. (in press).

Beauzamy, B., 1992. Products of polynomials and a priori estimates for coefficients in polynomial decompositions: a sharp result. J. Symbolic Comput. 13 (5), 463–472.

Belabas, K., Hanrot, G., Zimmermann, P., 2001. Tuning and generalizing van Hoeij's algorithm, Rapport de recherche 4124, INRIA.

Berlekamp, E.R., 1970. Factoring polynomials over large finite fields. Math. Comp. 24, 713–735.

Birkhoff, G., 1914. An elementary double inequality for the roots of an algebraic equation having greatest absolute value. Bull. AMS 14, 494–495.

Buchmann, J.A., Lenstra, Jr. H.W., 1994. Approximating rings of integers in number fields. J. Théor. Nombres Bordeaux 6 (2), 221–260.

Cerlienco, L., Mignotte, M., Piras, F., 1987. Computing the measure of a polynomial. J. Symbolic Comput. 4 (1), 21–33.

Cohen, H., 1993. A Course in Computational Algebraic Number Theory. Springer-Verlag.

Daberkow, M., Fieker, C., Klüners, J., Pohst, M., Roegner, K., Schörnig, M., Wildanger, K., 1997. KANT V4. J. Symbolic Comput. 24 (3–4), 267–283. Computational algebra and number theory (London, 1993).

Davenport, J.H., Mignotte, M., 1990. On finding the largest root of a polynomial. RAIRO Modél. Math. Anal. Numér. 24 (6), 693–696.

Fieker, C., Friedrichs, C., 2000. On reconstruction of algebraic numbers. In: Algorithmic Number Theory (Leiden, 2000). LNCS, vol. 1838. Springer, pp. 285–296.

Ford, D., Pauli, S., Roblot, X.-F., 2002. A fast algorithm for polynomial factorization over $\mathbb{Q}_p$. J. Théor. Nombres Bordeaux 14 (1), 151–169.

Gourdon, X., 1993. Algorithmique du théorème fondamental de l'algèbre, Rapport de recherche 1852, INRIA.

Householder, A.S., 1970. The Numerical Treatment of a Single Nonlinear Equation. International Series in Pure and Applied Mathematics. McGraw-Hill Book Co., New York.

Knuth, D.E., 1969. The Art of Computer Programming. Seminumerical Algorithms, vol. 2. Addison-Wesley.

Koy, H., Schnorr, C.-P., 2001a. Segment LLL-reduction of lattice bases. In: CaLC. LNCS, vol. 2146. Springer, pp. 67–80.

Koy, H., Schnorr, C.-P., 2001b. Segment LLL-reduction with floating point orthogonalization. In: CaLC. LNCS, vol. 2146. Springer, pp. 81–96.

Lenstra, A.K., 1982. Lattices and Factorization of Polynomials Over Algebraic Number Fields, (Berlin). LNCS, vol. 144. Springer, Berlin, pp. 32–39.

Lenstra, A.K., Lenstra, Jr., H.W., Lovász, L., 1982. Factoring polynomials with rational coefficients. Math. Ann. 261 (4), 515–534.

Mignotte, M., 1974. An inequality about factors of polynomials. Math. Comp. 28, 1153–1157.

Neukirch, J., 1986. Class Field Theory. Springer-Verlag, Berlin.

Pohst, M., 1993. Computational Algebraic Number Theory. DMV Seminar, vol. 21. Birkhäuser, Basel.

Pohst, M., Zassenhaus, H., 1989. Algorithmic Algebraic Number Theory. Encyclopedia of Mathematics and its Applications, vol. 30. Cambridge University Press, Cambridge.

Roblot, X.-F., 2002. Polynomial factorization algorithms over number fields, J. Symbolic Comput. (in press).

Schnorr, C.-P., Euchner, M., 1994. Lattice basis reduction: improved practical algorithms and solving subset sum problems. Math. Program. Ser. A 66 (2), 181–199.

Shoup, V., 2003. NTL: A library for doing Number Theory.
Available from http://www.shoup.net/ntl/.

Stoer, J., Bulirsch, R., 2002. Introduction to Numerical Analysis, third ed, Texts in Applied Mathematics, vol. 12. Springer-Verlag, New York (Translated from the German by R. Bartels, W. Gautschi and C. Witzgall).

The PARI group, 2003. PARI/GP, version 2.2.6, Bordeaux.
Available from http://www.parigp-home.de.

van Hoeij, M., 2002. Factoring polynomials and the knapsack problem. J. Number Theory 95 (2), 167–189.

Zassenhaus, H., 1969. On Hensel factorization I. J. Number Theory 291–311.

Zimmermann, P., 1996. Polynomial factorization challenges: a collection of polynomials difficult to factor. Available from http://www.loria.fr/~zimmerma/mupad/.